ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Сәтбаев университеті

Кибернетика және ақпараттық технологиялар институты

Киберқауіпсіздік, ақпаратты өңдеу және сақтау кафедрасы

Оқасова Алина Еркінқызы

Веб-қосымшаларды әзірлеу «Нақты уақыт қосымшаларына арналған платформа»

ДИПЛОМДЫҚ ЖОБА

5В070300–«Ақпараттық жүйелер» мамандығы

Алматы 2020

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Сәтбаев университеті

Кибернетика және ақпараттық технологиялар институты

Киберқауіпсіздік, ақпаратты өңдеу және сақтау кафедрасы

ҚОРҒАУҒА ЖІБЕРІЛДІ
КАӨжС кафедра меңгерушісі,
тех.ғыл.канд., ассоц.
профессор
_____Н.А.Сейлова
«_____» _____2020ж.

ДИПЛОМДЫҚ ЖОБА

Тақырыбы: Веб-қосымшаларды әзірлеу «Нақты уақыт қосымшаларына арналған платформа»

5В070300 – «Ақпараттық жүйелер» мамандығы бойынша

Орындаған: Оқасова А.Е.
Пікір беруші: Ғылыми жетекші:
_____ Рысқұлбек Е.А, тьютор
«_____» _____ 2020 ж. «_____» _____ 2020 ж.

Алматы 2020

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Сәтбаев университеті

Кибернетика және ақпараттық технологиялар институты

Киберқауіпсіздік, ақпаратты өңдеу және сақтау кафедрасы

5В070300 – «Ақпараттық жүйелер» мамандығы

БЕКІТЕМІН
КАӨжС кафедра меңгерушісі,
тех.ғыл.канд., ассоц.
профессор
_____Н.А.Сейлова
«_____» _____ 2020 ж.

Дипломдық жобаны орындауға
ТАПСЫРМА

Білім алушы: Оқасова Алина Еркінқызы
Тақырыбы:« Нақты уақыт қосымшаларына арналған платформа»
Университет Ректоры: 2020 *жылғы* «27» қаңтар №762-б бұйрығымен бекітілген
Аяқталған жобаны тапсыру мерзімі: 2020 *жылы* «29» мамыр
Дипломдық жобаның бастапқы берілістері: диплом алдындағы практикалық жұмыс қорытындысы, тақырып бойынша әдебиеттерге шолу нәтижелері, теориялық мәліметтердің жиыны
Дипломдық жобада қарастырылатын мәселелердің тізімі:
а) қойылған мәселенің қазіргі жағдайын пайымдау
б) ақпараттық қамтаманы құру
Сызбалық материалдар тізімі: Power Point бағдарламасындағы слайдтар
Сызба материалдар: 15 слайдпен көрсетілген
Ұсынылатын негізгі әдебиет: 9 *атау*

Дипломдық жобаны дайындау

КЕСТЕСІ

Бөлімдер атауы, қарастырылатын мәселелер тізімі	Ғылыми жетекші мен кеңесшілерге көрсету мерзімдері	Ескерту
Мәселенің қазіргі жағдайына шолу және оны талдау	10.01.2020 – 08.02.2020	
Ақпараттық қамтаманы құру	09.02.2020 – 10.03.2020	
Программалық қамтаманы құру	11.03.2020 – 28.04.2020	

Дипломдық жобасының бөлімдерінің кеңесшілері мен норма бақылаушыларының аяқталған жобаға қойған **қолтаңбалары**

Бөлімдер атауы	**Кеңесшілер, аты, әкесінің аты, тегі (ғылыми дәрежесі,атағы)**	**Қол қойылған күні**	Қолы
Норма бақылаушы	Дуйсенбаева А.Н., лектор, магистр		
Программалық қамтама	Кабдуллин М.А., ассистент		

Ғылыми жетекші_____Рысқұлбек Е.А.

Тапсырманы орындауға алған білім алушы_____Оқасова А.Е.

Күні «27» қаңтар 2020 ж.

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Сәтбаев университеті

Ғылыми жетекшінің пікірі

Дипломдық жоба

Оқасова Алина Еркінқызы

5В070300 – Ақпараттық жүйелер

Тақырыбы: «Нақты уақыт қосымшаларына арналған платформа»

Бұл дипломдық жоба өзінің логикалық құрылымымен ерекшеленген. Түсіндірме жобаның құрамы кіріспеден, 3 бөлімнен, қорытындыдан, әдебиеттер тізімінен және қосымшадан тұрады.

Менің пікірімше, диплом жобалаушы алдына қойылған тапсырманы толығымен орындады және кейінгі технологияларын меңгергендігін көрсетті.

Жалпы дипломдық жоба профессионалдық деңгейде орындалған. Түсіндірме жазба сауатты бейнеленген, жоба бойынша барлық қажетті ақпараттар бар.

Кемшілік ретінде кейбір шағын стилистикалық қателерді атап кетуге болады.

Жоғарыда айтылғандарға байланысты, дипломдық жоба 5В070300 – «Ақпараттық жүйелер» мамандығының бітіру жұмыстарына қойылатын талаптарына сәйкес және дипломдық жобаны қорғауға жіберіле алады, ал оның авторы Оқасова Алина Еркінқызы бакалавр академиялық дәрежесін алуға лайықты деп есептеймін.

Ғылыми жетекші
Тьютор Рысқұлбек Е.А.
«___» _____ 2020 ж.

Протокол анализа Отчета подобия Научным руководителем

Заявляю, что я ознакомился(-ась) с Полным отчетом подобия, который был сгенерирован Системой выявления и предотвращения плагиата в отношении работы:

Автор: Окасова Алина
Название: Платформа для приложении реального времени
Координатор: Ерсұлтан Рыскұлбек

Коэффициент подобия 1:1,8

Коэффициент подобия 2:0

Замена букв:2
Интервалы:0
Микропробелы:0
Белые знаки: 0

После анализа Отчета подобия констатирую следующее:

 обнаруженные в работе заимствования являются добросовестными и не обладают признаками плагиата. В связи с чем, признаю работу самостоятельной и допускаю ее к защите;

 обнаруженные в работе заимствования не обладают признаками плагиата, но их чрезмерное количество вызывает сомнения в отношении ценности работы по существу и отсутствием самостоятельности ее автора. В связи с чем, работа должна быть вновь отредактирована с целью ограничения заимствований;

 обнаруженные в работе заимствования являются недобросовестными и обладают признаками плагиата, или в ней содержатся преднамеренные искажения текста, указывающие на попытки сокрытия недобросовестных заимствований. В связи с чем, не допускаю работу к защите.

Обоснование:
……
……

…… ……
Дата *Подпись Научного руководителя*

Протокол анализа Отчета подобия

заведующего кафедрой / начальника структурного подразделения

Заведующий кафедрой / начальник структурного подразделения заявляет, что ознакомился(-ась) с Полным отчетом подобия, который был сгенерирован Системой выявления и предотвращения плагиата в отношении работы:

Автор: Окасова Алина

Название: Платформа для приложении реального времени

Координатор: Ерсұлтан Раскұлбек

Коэффициент подобия 1:1,8

Коэффициент подобия 2:0

Замена букв:2

Интервалы:0

Микропробелы:0

Белые знаки:0

После анализа отчета подобия заведующий кафедрой / начальник структурного подразделения констатирует следующее:

☐обнаруженные в работе заимствования являются добросовестными и не обладают признаками плагиата. В связи с чем, работа признается самостоятельной и допускается к защите;

☐обнаруженные в работе заимствования не обладают признаками плагиата, но их чрезмерное количество вызывает сомнения в отношении ценности работы по существу и отсутствием самостоятельности ее автора. В связи с чем, работа должна быть вновь отредактирована с целью ограничения заимствований;

☐обнаруженные в работе заимствования являются недобросовестными и обладают признаками плагиата, или в ней содержатся преднамеренные искажения текста, указывающие на попытки сокрытия недобросовестных заимствований. В связи с чем, работа не допускается к защите.

Обоснование:

……
……
……
……
……
……
……

…… ……

Дата *Подпись заведующего кафедрой /*

начальника структурного подразделения

Окончательное решение в отношении допуска к защите, включая обоснование:

……
……
……
……
……
……
……

…… ……

Дата *Подпись заведующего кафедрой /*

начальника структурного подразделения

АҢДАТПА

Әлеуметтік желілерге қосымшалардың қарқынды өсуінен кейін үлкен көлемде деректерді басқару үшін мазмұнды басқару жүйелері жасалды. Бұл дипломдық жобада осы жүйелердің бірі ретінде қызмет ететін қосымшаны жасау процесі егжей-тегжейлі сипатталған. Бағдарлама негіз ретінде ReactJS-ке негізделген. Жобада қолданылатын барлық ұғымдар мен технологиялар тиісті бөлімдерде сипатталған. Бұған клиент-сервер моделі, мазмұнды басқару жүйесі және құжат объектісінің моделі сияқты тиісті теориялық негіздер кіреді.

Дипломдық жобаның мақсаты нақты уақыттаға веб-қосымшаны, WebRTC технологиясын қолдана отырып, онлайн-кездесу функциясын құру болып табылады. Дипломдық жобаны орындау барысында ұқсас заманауи жүйелерге шолу жасалды. Жүйені жасау құрылғысы ретінде заманауи React кітапханасы таңдалды. Жаңа технология ретінде WebRTC технологиясы қарастырылды.

Берілген жобаның нәтижесі ретінде нақты уақыттағы веб-қосымшаны қажет ететін кез-келген салада қолдануға болады.

АННОТАЦИЯ

После быстрого роста приложений для социальных сетей были разработаны системы управления контентом для управления большими объемами данных. Этот дипломный проект подробно описывает процесс создания веб-приложения, которое служит одной из этих систем. Программа основана на ReactJS. Все концепции и технологии, используемые в проекте, описаны в соответствующих разделах. Это включает в себя соответствующие теоретические основы, такие как модель клиент-сервер, система управления контентом и объектная модель документа.

Целью дипломного проекта является создание в режиме реального времени веб-приложения, с функции онлайн-встречи с использованием технологии WebRTC. В ходе работы был сделан обзор подобных современных систем. В качестве устройства для создания системы была выбрана современная библиотека React, технология WebRTC которая считается новой технологией.

В результате этого проекта вы можете использовать веб-приложение в реальном времени в любой области, где оно требуется.

ABSTRACT

After the rapid growth of applications for social networks, content management systems have been developed to manage large amounts of data. This thesis describes in detail the process of creating a web application that serves as one of these systems. The program is based on ReactJS. All concepts and technologies used in the project are described in the relevant sections. This includes relevant theoretical foundations, such as a client-server model, a content management system, and a document object model.

The aim of the thesis is to create a real-time web application, with the function of an online meeting using WebRTC technology. In the course of the work, an overview of such modern systems was made. As a device for creating the system, the modern React library was chosen and WebRTC technology which is considered a new technology.

As a result of this work, you can use the web application in real-time in any area where it is required.

МАЗМҰНЫ

КІРІСПЕ

Осыдан оншақты жылдар бұрын ғана қоғамды алдыңғы қатарлы технологиядар дәуірі туралы ой таңғалдыратын. Бірақ қазір бұл арман емес, бұл шындық. Қазіргі кезде Ғаламтор адамзаттың ең қарқынды дамып келе жатқан ақпарат алмасу ортасы болып табылады. Ғаламтордың толықтай және жартылай көрінісі болып табылмайтын қызметтің қандай да бір саласын атап айту қиын.

Ол талай жылдарда технологиялық өзгерістерге ұшырап, қазір біздің күнделікті өміріміздің айырымас бөлігі болып табылады. Әлеуметтік желілер, Веб-қосымшалар, онлайн хабар алмасу, қоңырау шалу – қарапайым қолданушы күніне бірнеше рет орындай алатын міндеттер.

Бұл ауысулар жылдам қарқынды болғандықтан пайдаланушыларды әрқашан тиімділік, ыңғайлылық қажеттілігі туралы ойландырады.

Алайда, мұның бәрін веб-қосымшаларсыз елестету қиын болар еді. Веб-қосымшалар ол клиентпен сервер арасындағы байланыс, яғни клиент браузер арқылы веб-сервермен өзара әрекетеседі. Мұнда браузер клиент, ал веб-сервер сервер болып табылады. Технологиялар әрдайым өзгерістерге бейім болғаннан, веб-қосымшаларды да айналып өтпеді. Қолданушылардың күту уақытын қысқарту, желі трафигін үнемдеу және басқада да өзекті мәселелерге шешім болып SPA болып табылады. Себебі бұнда веб-сайттан қарағанда SPA қолданушылармен интерактивті байланыс керек ақпарат қана жаңартылады да жұмыстың уақытын үнемдейді. Яғни нақты уақыттағы қосымша қазіргі кездегі тиімдісі, қолданушылардың қалауы болып табылады.

Осы дипломдық жобада нақты уақыттағы қосымша жасауға қолданушылар арасындағы интерактивті байланыс орнатуды орындаймыз. Бұл веб-қосымша қолданушыларға арнайы нақты уақыттағы байланыс яғни видео-чат, ақпарат алмасу, қолданушылар ойын арқылы ақпарат алмасу ретінде көрсетілген үлкен жоба болады. Пайдаланушылар үшін қол жетімді деректер мен функциялар әр түрлі қол жеткізу құқықтарына негізделген. Барлық мәліметтер қатысты қосымша сервер жағында орналасқан дерекқорда сақталады. Сервер веб-браузері жіберілген және кері жіберілетін барлық деректерді қабылдайды және барлық сұралған деректерді өңдейді. Орындау үшін React, Node.js, WebRTC қолданылады.

1 ҚАЗІРГІ ЗАМАНДАҒЫ ЖАҒДАЙҒА ШОЛУ

1.1 Қазіргі кездегі веб қосымшалар

Веб-бағдарлама жай ғана белгілеу тілдерін қолдана отырып, кейбір деректерді көрсетуге арналған кезден бастап едәуір дамыды. Бағдарламалық жасақтаманы пайдаланғысы келетін кез-келген машинада кез-келген бағдарламалық жасақтаманы орнату қажет болатын уақыт болды. Серверлік технологияларды, шолғыштарды және интернет жылдамдығын жақсарту SAAS құбылыстарын күшейтті. Бағдарламалық жасақтама құру ережесі түбегейлі өзгерді. Бүгінгі таңда жасалынған бағдарламалық жасақтаманың көпшілігі «кез-келген уақытта кез-келген машинада пайдалану философиясына» ескеріліп жасалады. SAAS қызметтері көбірек пайда болып, нарықтағы кеңістік үшін бәсекеге түсуде. Бұл мүмкіндіктерге бай веб-қосымшаны құру үшін қосалқы өнеркәсіптің қажеттілігін қолдау үшін бірнеше технологиялар мен негіздердің пайда болуына себеп болды.

SAAS провайдерінің көпшілігі нақты уақыттағы веб-қосымшаның қажеттілігін түсіне бастады. Нақты уақыттағы веб-қосымшаларды жасау кеңістігіндегі тартымдылық нақты уақыт режиміндегі веб-қосымшаны құруға қолдау көрсететін түрлі құрылымдар мен технологиялар арқылы көрінеді. Енді веб-қосымшасы клиент пен сервер арасындағы дәстүрлі сұраныс-жауап циклімен шектелмейді. Веб-қосымшасы клиент пен сервер арасындағы нақты уақыттағы деректерді синхрондауды қолдау үшін мәліметтер базасынан кейбір мазмұнды көрсетуден белгілі бір ұзақ жолды өтті. Нақты уақыттағы веб-бағдарлама - бұл веб-дамудың болашағы. Қазіргі таңда SAAS қызметтері көбірек, нақты уақыт режимін қолдайды. Бірнеше жыл бұрын нақты уақыттағы веб-бағдарламада чат қосымшасы сияқты бірнеше жағдайларда ғана қолданған.

Соңғы жылдары вебке негізделген мультимедиялық қосымшалар өзінің әлеуеті мен мүмкіндіктеріне байланысты үлкен маңызға ие болды. Қазіргі нарықта көптеген қосымшалар қол жетімді және оларды көптеген компаниялар пайдаланады.

1.2 Дипломдық жобаның тақырыбын таңдауды негіздеу

Республикамызда және басқада елдерде веб-қосымшаларға сұранысы көп. Қазіргі таңдағы жағдайға байланысты барлық жұмыс, қызмет онлайн режиміне ауысты. Соған байланысты еліміздің осындай онлайн режимге ауысуды біздің ғаламтор желісі дайын емес екенін көрсетті. Осыған орай еліміздің осыдан бастап ақпараттық желілерге бөлінер назары мол болмақ. Яғни веб-қосымшалардың интерактивтілігі, онлайн аудио, видео байланыс. Оқушылар, студенттер, жұмысшыларға бұл процесстерді бейімдеу, жалпы қолданысқа енгізу.

Онлайн-коммуникация қазіргі әлемде аса маңызды болып келеді. Біз оны күн сайын отбасымызбен, достарымызбен қарым-қатынас жасау және біздің команда мүшелерімен немесе клиенттермен іскерлік ортада қарым-қатынас жасау үшін пайдаланамыз. Үй кеңсесі немесе тіпті қашықтан жұмыс көптеген кәсіпорындар мен олардың қызметкерлері арасында көп танымал. Екі тарап қашықтықтан жұмысынан пайда табуы мүмкін. Қызметкерлер оларға күн сайын жұмысқа баруына тура келмейді және олардың жақындарымен уақыт көп өткізуге болады. Екінші жағынан, кәсіпорындар ақшаны үнемдей алады, өйткені олар кеңсе кеңістігін ұстап тұру үшін тұрақты қызметкерлер болмайды. Осы үрдіспен команданың қалған бөлігімен қарым-қатынастың жоғары сапасы мен сенімді тәсіліне үлкен қажеттілік туады.

Осы тенденцияның арқасында команданың қалған бөліктерімен сапалы және сенімді қарым-қатынас қажет.

Веб-қосымшалар үлкен әртүрлі мақсаттар үшін қолданылады, мысалы, тағайындау жүйелері, жедел хабар алмасу қызметтері, сауда, ойын-сауық және тіпті кейбір елдерде демократиялық шешім қабылдау.

Жылдам жұмыс істейтін құрылғылармен қатар жылдам қосымшалар қажет. Веб пен мобильді құрылғыларда қолдануға болатын қосымшалар көп. Әр түрлі қосымшаларды жасау үшін бірнеше JavaScript негіздері мен кітапханалар қолданылады. Қазіргі уақытта өндірісте ReactJS, AngularJS, EmberJS, MeteorJS, VueJS, KnockoutJS және басқалары бар. ReactJS - қосымшаларды жетілдіруге арналған құралдардың бірі. React - бұл Facebook-те жасалынған әйгілі кең танымал JavaScript-кітапхана. React қолданушының интерактивті интерфейсін құруды жеңілдетеді. Ол әр күйге нақты компоненттерді ұсыну арқылы тиімді түрде жаңартылады және қосымшадағы деректерді өзгертеді. ReactJS-те әр компонент өзінің күйін басқарады және оларды қолданушы интерфейстеріне құрады. Бұл JavaScript-тегі шаблондардың орнына компоненттер ұғымы, көптеген мәліметтер қосымшаларға оңай ауысып, DOM-дан тыс күйге түседі. Node React бағдарламасын серверде де көрсетуге болады. Мобильді қосымшаларды құру үшін веб-бағдарламалармен қатар React Native бағдарламасын да пайдалануға болады.

Бұл жобаның негізінде жаңа технологияларды қарастыра отырып қолданушылар арасындағы нақты уақытта жұмыс жасайтын веб-қосымшаны құру болып табылады. Бұл жобаның барысында жаңа технологияларды үйреніп практикада қолданылмақ. Дипломдық жобаның мақсаты - JavaScript негізінде ReactJS кітапханасында терең зерттеу жүргізу, ReactJS кітапханасы туралы терең түсінік беру.

1.3 Есептің қойылымы

Бұл дипломдық жобаның негізгі мақсаты – нақты уақытағы веб-қосымшасын құру болып табылады. Веб-қосымша жұмысының қызметін

жүйелендіріп қолданушыларға қолайлы және тез қызмет көрсететін жүйе жасау қажет.

Жүйеде келесідей функциялар орындалуы керек:

– Екі қолданушылар арасындағы байланыс орнату, ақпарат алмасу сокеттермен жұмыс.

– Қолданушылар арасындағы видео байланыс жаңа технология WebRTC қолдану.

– Қолданушыларға арналған интерфейс жасау үшін React кітапханасын қолдану.

– Веб-қосымшада орындалатын ойын жасау.

– Қолданушылар арасындағы жасалған ойын арқылы жұмыс жасалуы, байланыс орнатылуы керек.

Жалпы бұл жобада Веб-қосымша құру болып табылады. Болашақта бұл веб-қосымша қолданушыларға тиімді әрі пайдалы болмақ. Себебі нақты уақыттағы байланыс, яғни екі қолданушы арасындағы байланыс, ақпарат алмасу. Сонымен қатар көптеген басқада функциялар рейтинг, көптеген жаңа материалдар, ғаламтордың күрделі, үлкен порталы болмақ.

2 АҚПАРАТТЫҚ ҚОСЫМШАНЫ ЖОБАЛАУ

2.1 Программалық ақпараттық құралдарды таңдау

React – бұл композиционды қолданушы интерфейсін құруға арналған JavaScript кітапханасы. Оның көмегімен сіз интерфейсті нақты уақытта жаңартуды қажет ететін әртүрлі бір беттік қосымшаларды жасай аласыз, мысалы чат бағдарламасы. React-і дамыту 2011 ж. Аяғында басталып, 2013 жылдың жазында Facebook Inc. шығарды және ашық түрде ұсынылды. Ол қазір ең көп қолданылатын JavaScript кітапханаларының бірі болып табылады және қазіргі уақытта Инстаграм, Facebook және Spotify-ті қосу үшін пайдаланылады.

Әзірлеушілер мен инженерлер React-ті таңдауда, өйткені бұл көбінесе өнімді дамытуға көп уақыт бөліге мүмкіндік береді, ал құрылымды зерттеуге, күресуге және оқуға аз уақыт бөлуге болады.

React қосымшасы – бұл әрқайсысы бір көріністі білдіретін жеке компоненттер жиынтығы. Көріністің әр жеке компонентінің идеясы өнімді жасау кезінде итерацияны жеңілдетеді, өйткені жеке көрініске немесе компонентке өзгерістер енгізу үшін бүкіл жүйені қарастырудың қажеті жоқ. Бағдарлама React көмегімен жасалған кезде, код әдетте болжалды, себебі React өзгермелі императивті DOM API-ны декларативті түрде орайды, бұл абстракция деңгейін арттырады және бағдарламалау моделін жеңілдетеді. Сонымен қатар, React бағдарламасымен жасалған қосымшаны масштабтау оңайырақ.

Кейбір басқа құрылымды үйренуге көп уақыт кететін JavaScript кітапханалардан айырмашылығы, React қосымшаны құруға көп күш жұмсамайды.

React көптеген күшті жақтардан тұрады. Жақсы оқулуы - React-тің ең күшті жақтарының бірі. Онымен таныс емес адамдар үшін де оқу оңай. Басқа жақтаулар раманың өзі туралы көптеген тұжырымдамаларды зерттеуді қажет етеді, бірақ тілдің негіздерін ескермей, керісінше жасайды.

ReactJS-те қосымшалардағы күйлер мен қабаттар арасында бір бағытты мәліметтер ағыны бар. Бұл деректер қосымшалар күйлері мен қабаттар арасында бір бағытта берілетінін білдіреді. DOM жаңартуларын балама фреймворкалармен салыстырғанда тезірек жаңартады және бұл әлдеқайда аз кітапхана. DOM құжат нысаны моделін білдіреді. Осылайша, жұмысты орындау үшін құралдарды таңдау оңай.

React компоненттері көбінесе JSX деп аталатын JavaScript Extended көмегімен жазылады. JSX - бұл файлдарды бөлудің орнына логиканы да, түзетуді де бір файлға біріктірудің тәсілі. Реакция JSX қолдануды қажет етпейді, бірақ көптеген адамдар JavaScript ішіндегі UI-мен жұмыс істегенде пайдалы деп санайды, өйткені ол синтаксис сияқты HTML қолданады. JSX шаблондау тілі сияқты көрінеді, бірақ JavaScript-тің барлық мүмкіндіктерімен бірге, JSX-ті өте қуатты етеді.

Реакция компоненттерге негізделген. Компонент - бұл барлық қолданбада қолдануға болатын қайта пайдалануға болатын кодтың бөлігі. Ол сіздің

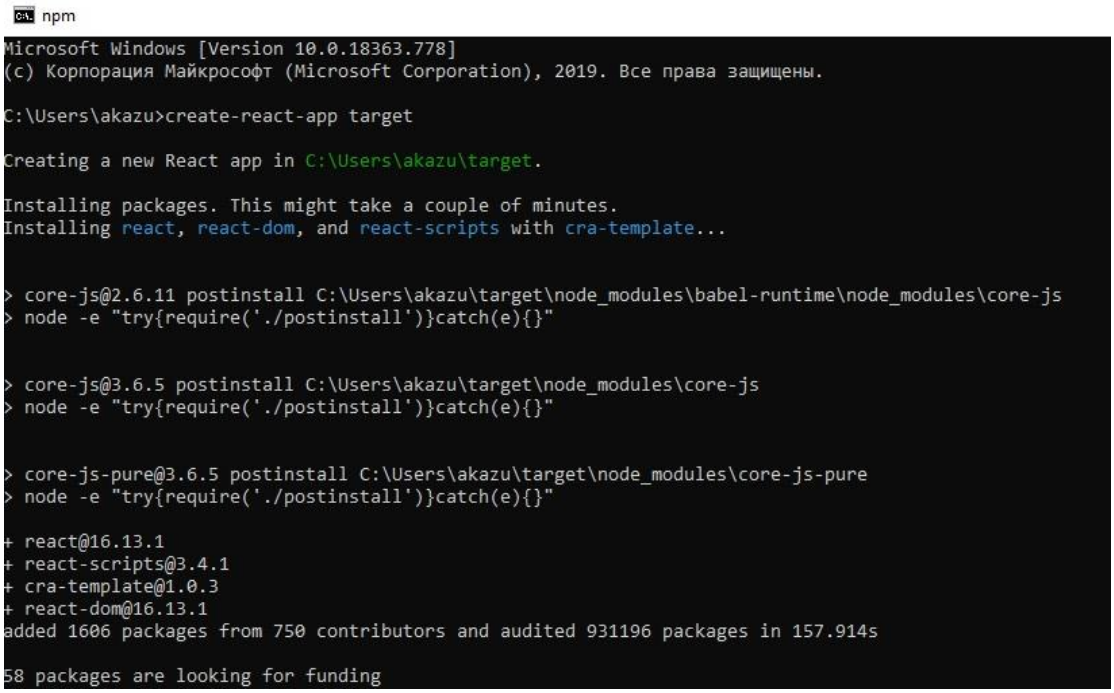
сайтыңыздағы бір нәрсе қалай көрінетінін және оның мінез-құлқын анықтай алады. Реакция компоненттері кодтың кіші бөліктерінен тұрады

Реакция элементтері өзгермейтін болып табылады. Бір рет жасалған мағынаны өзгерту мүмкін емес. Ресми реакция құжаттамасында элемент былай сипатталған: «элемент фильмдегі жалғыз кадрға ұқсас: ол UI интерфейсін белгілі бір уақытта көрсетеді».

UI интерфейсін жаңа элемент құру және оны бетке көрсету арқылы жаңартуға болады, бірақ түпнұсқа элемент әрқашан өзгермейді, себебі элементтер өзгермейді. UI элементтері жаңартылған кезде React.DOM тек оның қажетті бөліктерін жаңартады. Ол жаңа элементті және оның балаларын алдыңғы элементпен салыстырады және тек өзгерген нәрсені жаңартады. Яғни, минималды аз әрекет жасау арқылы жаңартуды жүргізеді.

Nodejs орнату үшін nodejs.org деп аталатын веб-сайтқа кіру керек және әртүрлі операциялық жүйелер үшін бірнеше жүктеу қондырғысы бар. Біріншіден, ол nodejs орнатады. Nodejs веб-серверін құруға мүмкіндік береді, осылайша React компоненттері жергілікті түрде қолданыла алады және интернетте тікелей орналастырыла алады. Ол сонымен қатар npm деп аталатын түйін пакетін басқарушыны орнатады, ол бізге бағдарламаларға React сияқты үшінші тарап модульдерін орнатуға мүмкіндік береді.

2.1 суретте кейбір бастапқы орнатудың скриншоты көрсетілген:

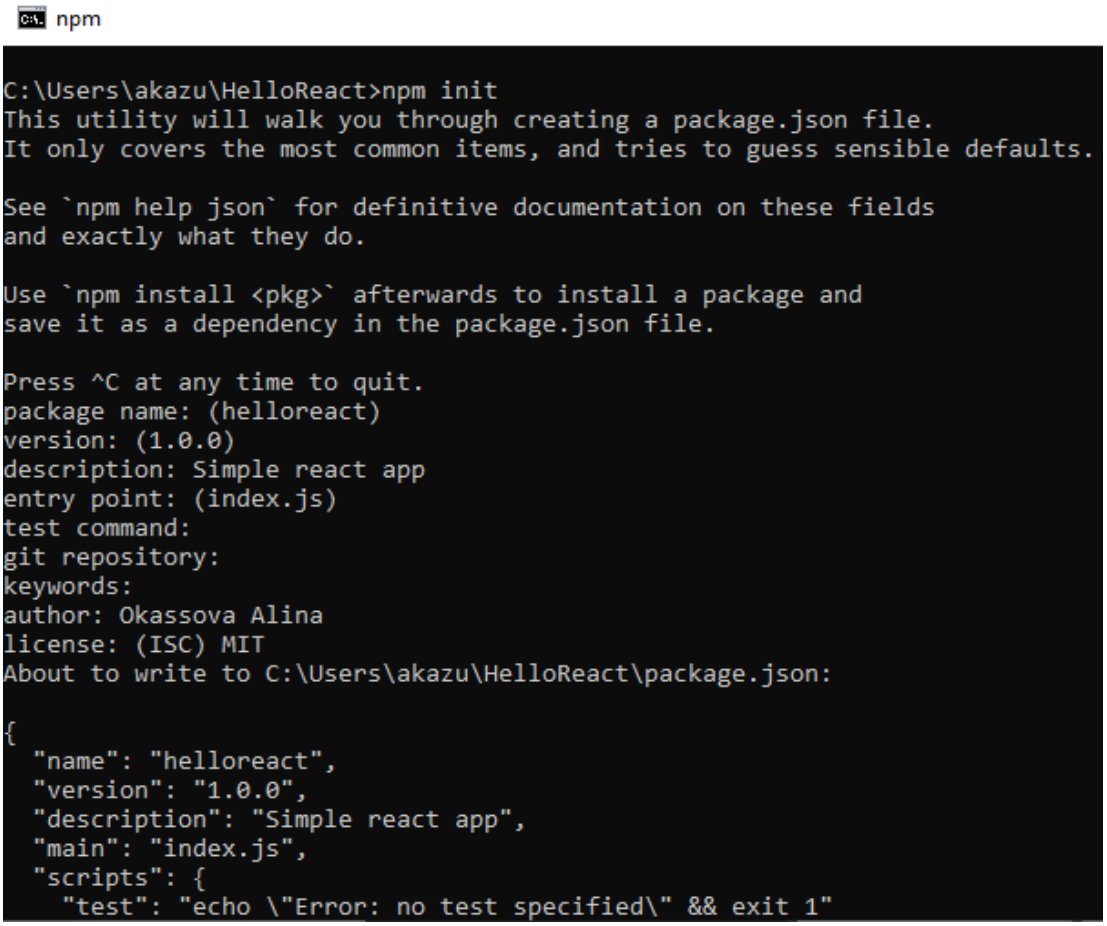


2.1 Сурет – Бастапқы орнату

Nodejs және npm пакет менеджері – ReactJS әзірлеуді орнатудың негізгі талаптары. Түйін және npm екеуі түйіннің веб-сайтынан жүктеу кезінде жинақталады.

Node.js – бұл асинхронды оқиғаға негізделген JavaScript жұмыс уақыты, оған JavaScript тілінде жазылған бағдарламаны іске қосуға арналған барлық нәрсе кіреді. Оның пайда болуы түпнұсқа JavaScript-ті жасаушылар оны тек браузерде жұмыс істеуден бөлек бағдарлама ретінде компьютерде іске қосылатын нәрсеге дейін кеңейткен кезде пайда болды. Бұл JavaScript-ке тек веб-сайттармен әрекеттесуден гөрі көп нәрсе жасауға мүмкіндік береді. Енді Node көмегімен JavaScript Python, Java және C ++ сияқты дамыған тілдермен бірдей жұмыс жасай алады. Node.js-де оқиғаларға негізделген блоктау жоқ I / O моделі қолданылады, бұл оны өндеуге оңай әрі тиімді етеді. Сондықтан, Node кеңейтілетін желілік қосымшаларды құруға арналған. I / O жергілікті файлдарды оқудан және жазудан RESTful HTTP сұранысын API-ге дейін орындауға дейін өзгеруі мүмкін.

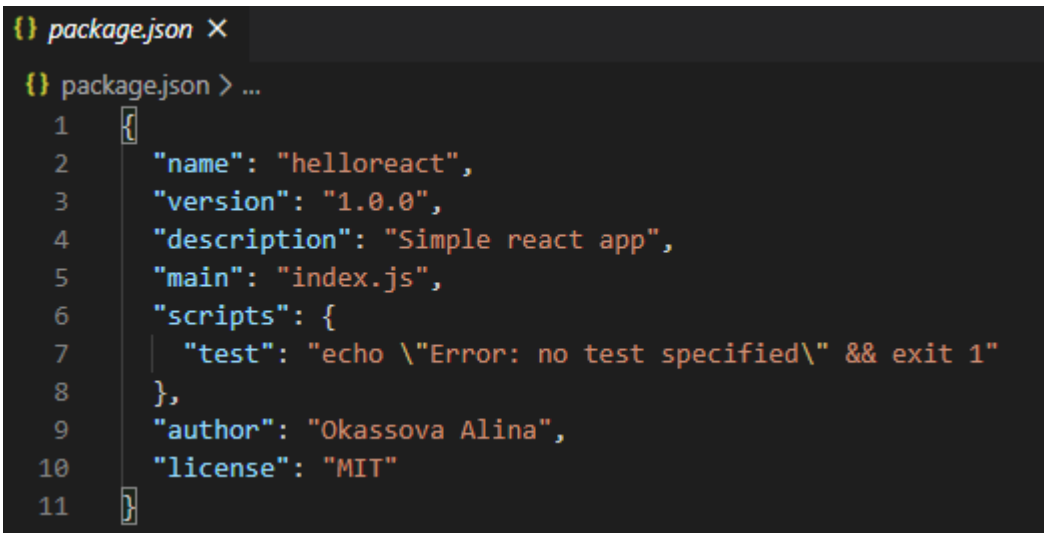
React-пен жұмыс істеу үшін алдын-ала қарапайым веб-сервер құру қажет. Веб-сервер болмаса, файлдарды шолғышта көруге мүмкіндік жоқ. Терминалды ашқаннан кейін жаңа түйіннің жобасын құру үшін npm-init қолдана аламыз. Мұнда алдымен HelloReact атты жобаны сақтау үшін жұмыс үстелінде папка жасаймыз. Терминалдан npm- init командасын орындаймыз және ол жобада бір файл жасайды. Ол дәл не істеп жатқандығы туралы аздап таныстырады, содан кейін бірнеше негізгі сұрақтарды қояды.



2.2 Сурет – Орнату мысалы

2.2 суретте көрсетілгендей, HelloReact жобасында npm- init пәрменін іске қосу бізге pack.json файлының баптауларынан өтуге мүмкіндік береді. Барлық сұрақтарға жауап бергеннен кейін, файл атауы, нұсқасы, авторы және лицензиясы, сәл файл файл пакет.json деп аталатын HelloReact жобасына файл жазатындығы туралы басып шығарады. Бұл файл тек түйін сервері үшін ғана емес, сонымен бірге тәуелділіктерді жою үшін де қолданылады.

Егер қазір VS Code редакторында HelloReact деп аталатын жаңа қалта ашылса, онда pack.json файлы терминалда көрсетілгендей көрінеді. 2.2 суретте pack.json файлы HelloReact Folder ішіндегі редакторда көрсетілген.

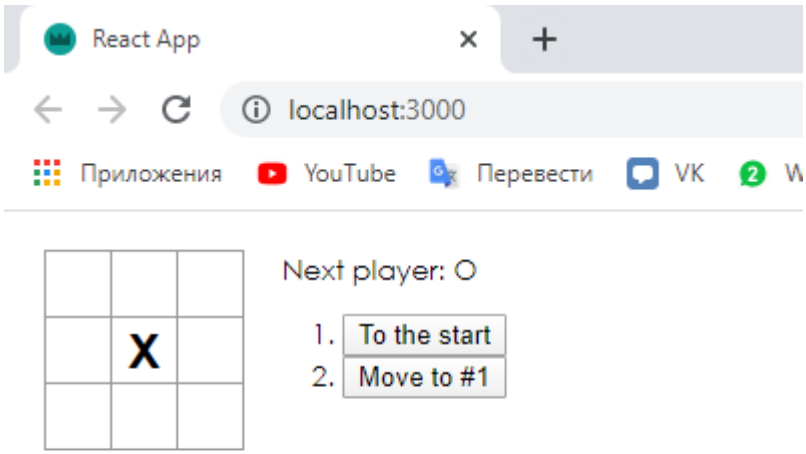


2.3 Сурет – VS Code редакторында көрсетілген

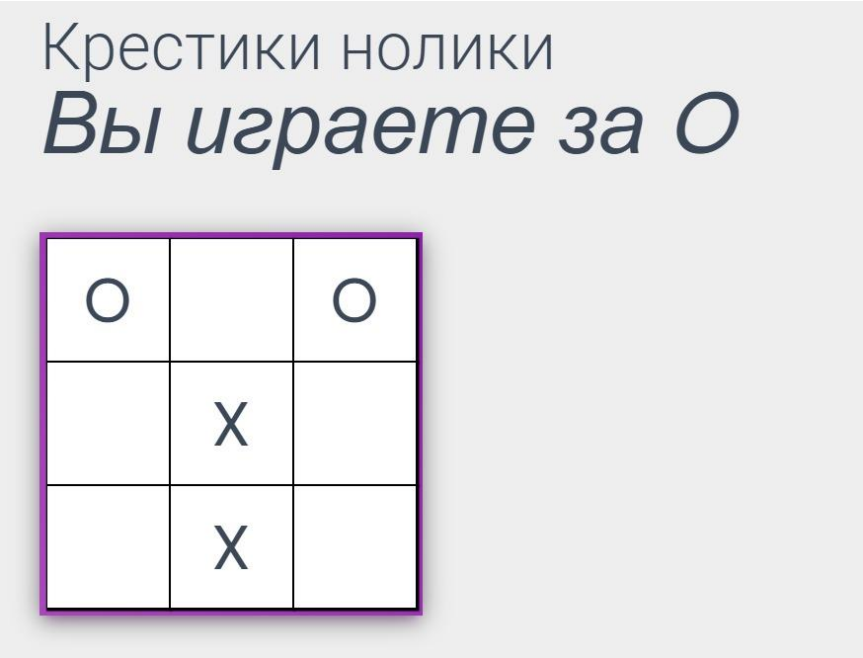
2.3 суретте көрсетілгендей, Package.json терминалда бұрын жасалған жауаптардан тұрады.

2.4 Сурет – Ойын кодының үзіндісі

2.4 суретте React-а жасалған ойын кодының үзіндісі. Екі қолданушы арасындағы байланыс WebSocket арқылы жасалады. Веб-сокеттер - бұл қарапайым TCP қосылымы арқылы толық дуплексті байланыс арнасын ұсынатын веб-байланыс протоколы. Қарапайым тілмен айтсақ, бұл технология тірі байланыстың барлық артықшылықтарын қолданатын қосымшаларды құруға мүмкіндік беретін минималды шығындармен клиент пен сервер арасында байланыс орнатуға мүмкіндік береді.



2.5 Сурет – Ойын интерфейсы



2.6 Сурет – Ойын интерфейсі

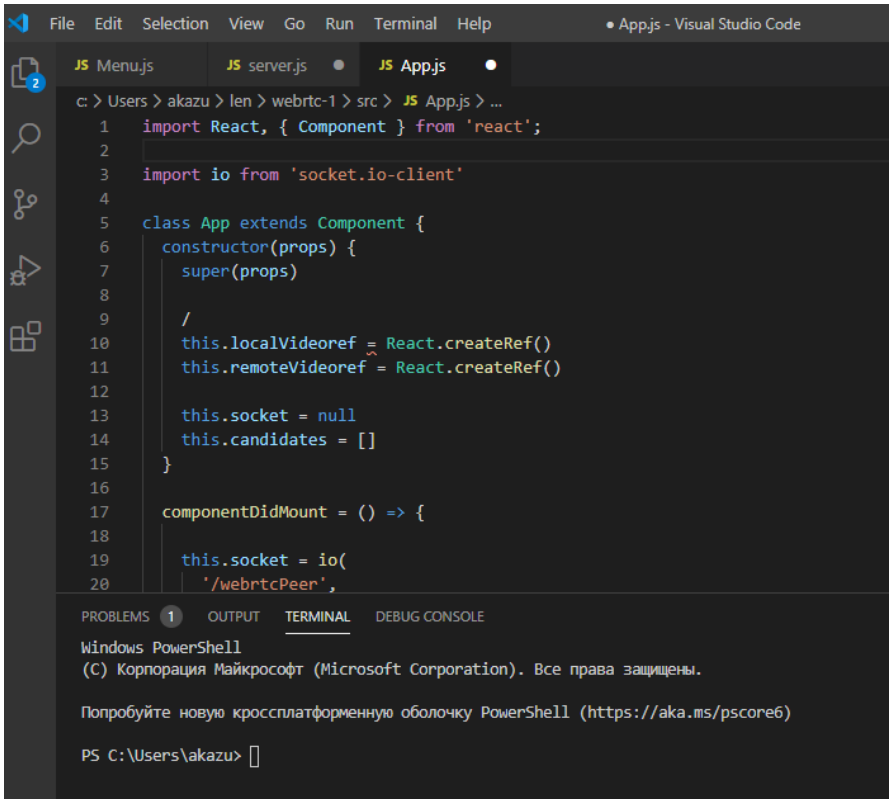
2.5 және 2.6 суретте React-і қолданып қолданушы интерфейсі көрсетілген. 2.4 суретіндегі ойынның нәтижесі шығарылған.

2.1.1 VS Code код редакторы

Visual Studio Code – Microsoft, Windows, Linux және macOS үшін бастапқы код редакторы. «Жеңіл» код редакторы ретінде ол веб және бұлтты қосымшалар үшін платформа құруға арналған. Жөндеу құралы, Git-пен жұмыс істеуге арналған құралдар, синтаксисті бөлектеу, Intelli AdSense және отқа төзімді құралдар бар. Оны күйге келтірудің көптеген нұсқалары бар: арнайы тақырыптар, пернелер тіркесімдері және конфигурация файлдары. Ол ашық бағдарламалық қамтамасыз ету ретінде ақысыз таратылады, бірақ дайын жинақ жеке меншік лицензиясы бойынша таратылады.

Visual Studio Code код өңдегішінің қарапайымдылығын әзірлеушілерге олардың негізгі түзету-құрастыру-түзету циклі үшін қажет нәрселермен үйлеседі. Ол кодты редакциялауға, навигация мен түсінуге жан-жақты қолдау көрсетеді, сонымен қатар жеңіл жөндеу, байытылған модель және қолда бар құралдармен оңай біріктіру.

Visual Studio коды ай сайын жаңа мүмкіндіктермен және қателерді түзетумен жаңартылады. Сіз оны Windows, macOS және Linux үшін Visual Studio Code сайтынан жүктей аласыз. Күн сайын соңғы шығарылымдарды алу үшін, Инсайдерлерді орнатыңыз.



2.7 Сурет - Visual Studio Code-да жазылған код

Код жазу кезінде тиімдірек болуға көмектеседі және Visual Studio Code-ді осындай пайдалы редакторға айналдырады 2.7 суретте көрсетілгендей:

– Редактордағы кодты жөндеу: редактордан кодты тексеріп, түзете аласыз.

– Нұсқаны басқару: Git көмегімен өзгерістерді бақылау үшін компьютердегі терминалға ауысудың қажеті жоқ.

– Интегралды терминал: Visual Studio Code көмегімен редактордан пәрмен жолдарын басқаруға болады.

2.2 Қолданған технологиялар

2.2.1 WebRTC нақты уақыттағы қосымшаға арналған технология

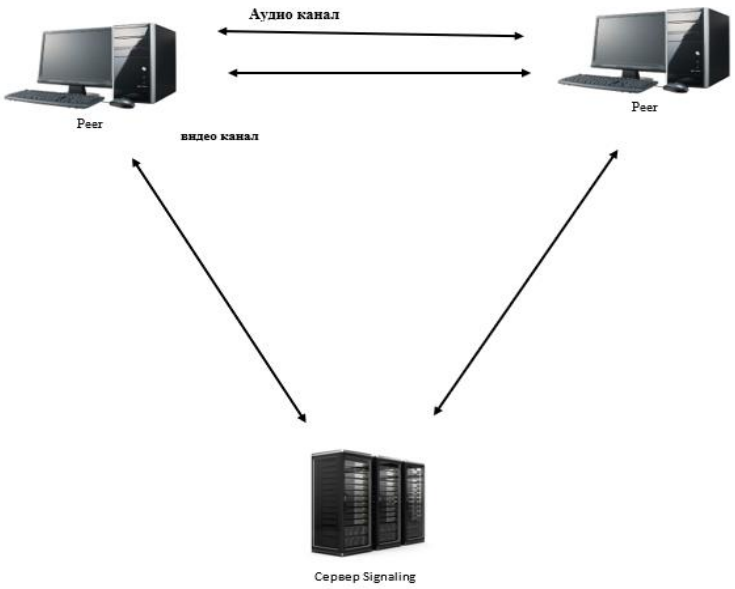
Бейнеконференция, цифрлық жиналыстар және скайп арқылы сұхбаттар уақыт пен ақшаның қысқаруы, сондай-ақ сапарлардың қоршаған ортаға тигізетін әсері түрінде жеңілдіктер беретін заманауи жұмысшылардың назарында болды. Алайда, қолданыстағы технологиялар пайдаланушының қолайлы интерфейсімен көп бөліктен тұратын қашықтықтан ынтымақтастықты тиімді қолдау үшін масштабтау, біріктіру және өзара әрекеттесу сияқты мәселелерді шешуге тырысуда. Дәстүрлі бейнеконференция жүйелерінде жүйелік ресурстарды (процессор, жад, диск және желі) ұшуға қолдануға бейімделу мүмкіндігі болмайды. Сонымен қатар, заманауи технологиялар жылдам нақты уақыттағы кері байланыс сияқты маңызды бөлшектерді жинай, жібере және көрсете алмайды, сондықтан интерактивті топтық жұмыс сияқты күрделі өзара әрекеттесу, әдетте, барлық серіктестердің қатысуын талап етеді. WebRTC-ті енгізу технологияның икемділігіне байланысты RTC-дегі көптеген мәселелерді шешу ретінде ерекшеленді, бұл жеңіл бөлісетін RTC-ге экранды бөлісу, экранды жазу, тақтаны бөлісу және Интернет арқылы видео / аудио чат сияқты ерекше функцияларсы бар мүмкіндік береді. TCP / IP қолдайтын вебинарлар мен веб-арналар сияқты веб-конференциялар 20 жыл бұрын бүкіләлемдік ғаламторда (WWW) қол жетімді болды. Содан бері технология бір бағыттан жартылай дуплексті және жартылай дуплексті мультимедиялық алмасуға көшті, оның ішінде өнімділікті арттыру және кідірісті азайту, деректердің жоғалуы, өткізу қабілеті мен процессор сияқты ресурстарды қазіргі заманғы бейне кодектер қолдана отырып пайдалану. Заманауи компьютерлер мен смартфондардағы қол жетімді сандық және бейнекамералар нақты уақыт режимінде Интернеттегі бейне байланыстарын үнемі арттыруды қолдайды. Нақты уақыттағы веб-байланыстың осы оң әрекеттері, жақсартылған сапасы және WWW-пен интеграциялану мүмкіндігі бар жаңа қосымшаларға сұраныстың артуы WebRTC-тің кейбір себептері болып табылады. WebRTC дегеніміз - бұл веб-қосымшаларға веб-браузер арқылы дауыстық қоңыраулар, видео чат және P2P файлдарын бөлісу сияқты түрлі RTC функцияларын қолдауға мүмкіндік беретін JavaScript әзірлеушілері үшін RTC API интерфейсін құру болып табылатын ашық бастапқы жоба. WebRTC API интерфейстері екі технологияның (HTML) және

JavaScript үйлесімінде жұмыс істейді. Шығарылғаннан кейін оны басқа байланыс технологияларымен, платформалармен біріктіруге күш салынды, бұл сіздің браузеріңізге, мобильді платформаларға және IoT құрылғыларына бай, жоғары сапалы RTC қосымшаларын жасауға мүмкіндік береді, олардың барлығы сөйлесуге мүмкіндік береді. хаттамалардың жалпы жиынтығы арқылы. WebRTC XSockets, Node.js, Vconect, OpenCV, Kinect, Kurento және басқа API сияқты басқа технологиялармен үйлесімде көптеген тартымды мүмкіндіктер мен функцияларды қамтамасыз етеді, соның ішінде нақты уақыт режимін анықтау және беттерді, эмоциялар, сөйлеу және қол қимылдарын тану. Осы технологияларды WebRTC-мен қалай дұрыс біріктіру керектігін үйрену RTC қызметтерін әртүрлі мақсаттарда ұсыну және енгізу тәсілдерін жетілдіре алады. Компаниялар арасындағы қашықтықтан жұмыс WebRTC енгізу әдеттегі жағдай болып табылады.

WebRTC көптеген тапсырмалар үшін қолданыла алатындығына қарамастан, оның нақты уақыт режиміндегі P2P мультимедиялық байланысы осы технологияның негізгі сатылым нүктесі ретінде қарастырылуы мүмкін. WebRTC-те екі клиент бір-бірімен веб-шолғыш арқылы байланысуы үшін алдымен олардың арасындағы байланыс туралы келіссөздер туралы ақпаратты жіберетін веб-сайтқа кіруі керек. Екіншіден, олардың веб-шолушылары қосылуды бастауға келісуі керек және олар брандмауэрдің қауіпсіздігі мен қорғауын айналып өтіп, Интернет арқылы бір-бірін қалай табуға болатындығын білуі керек. Ақыр соңында, олар барлық деректерді нақты уақыт режимінде беруі керек. Мультимедиялық құрылғыларға қол жеткізу үшін WebRTC GetUserMedia атты JavaScript нысанын қолданады.

Сеансты сипаттау протоколы (SDP) - ағынды медиа қосылымдарын олардың баптау параметрлерін сипаттай отырып конфигурациялауға көмектесетін стандарт. IETF 2006 жылдың шілдесінде IETF ұсынған стандарт ретінде қайта қарастырылған спецификацияны жариялады. SDP мультимедиялық байланыс сеанстарын, шақыруларды сипаттауға және тараптар арасындағы параметрлерді келісуге арналған. SDP нақты тасымалдағышты жеткізбейді, бірақ ағынның қандай медиа түрі мен басқа да онымен байланысты қасиеттері қолданылатын соңғы нүктелер арасында келісу үшін қолданылады.

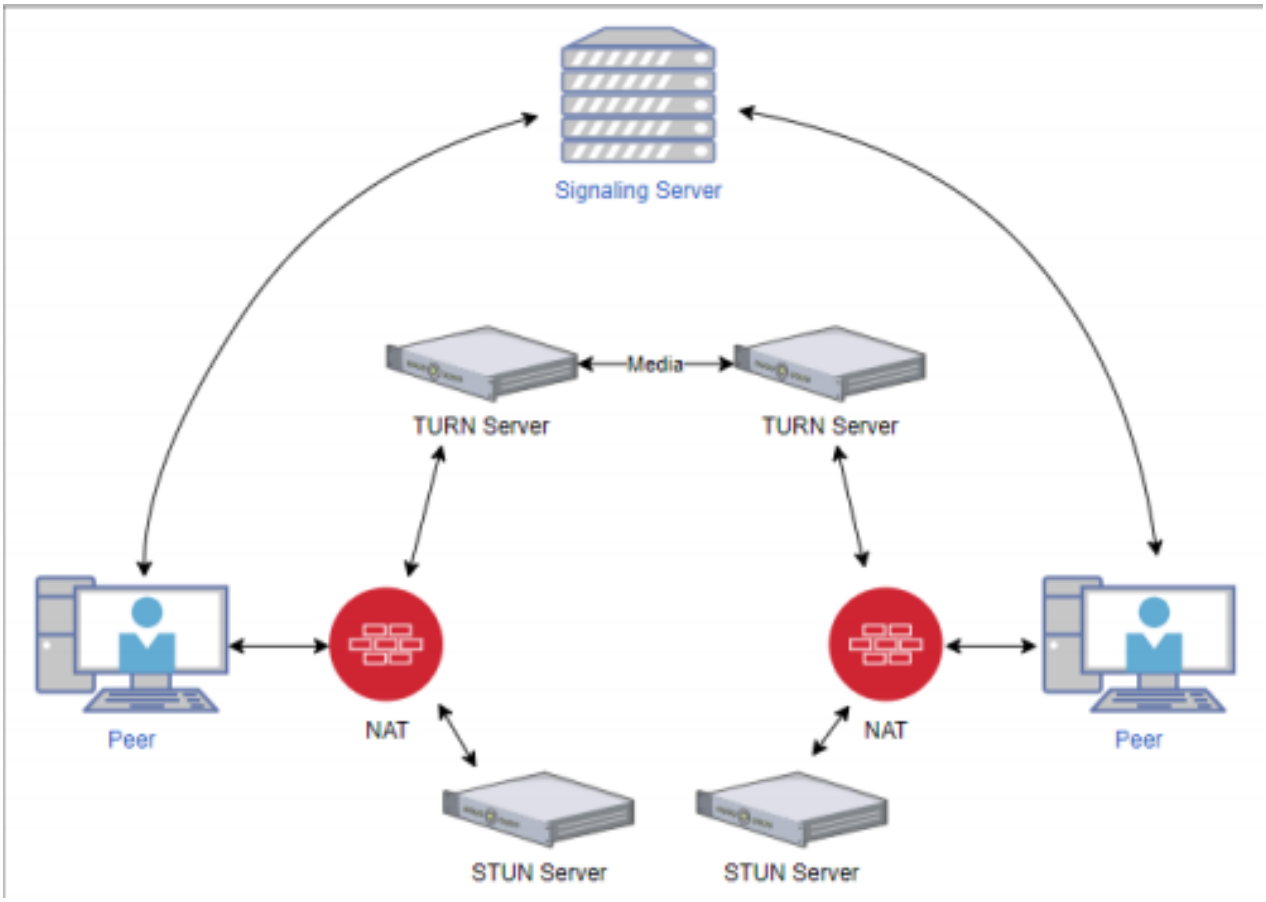
WebRTC-ке қойылатын талаптар әзірлеушінің көзқарасы бойынша WebRTC қосымшасын іске асыру үшін байланысуды қалайтын браузерлер арасында байланыс орнату қажет. WebRTC медиа-ұшақтың шешімін ұсынады, бірақ сигнал беруді жүзеге асырмайды, яғни бір клиенттен ақпараттар Peer-дан peer-ға қалай жіберілуі керек. Бұл Signaling деп аталады. Signaling көптеген жолдармен жүзеге асыруға болады, сондықтан әзірлеушілердің өзі бұл шешімді таңдауы керек. WebSockets көмегімен серверге тиімді шешім бола алады. Веб-сайттар шолғыш пен сервер арасында ашық екі жақты байланыс жасайды. Бұл браузер мен сервердің өзара әрекеті оқиғаларға негізделгенін білдіреді және мәлімет алмасу үшін әр уақытта серверден жауап сұраудың қажеті жоқ. Бұл өте жылдам және интерактивті байланысқа әкеледі, бұл сигналдық сервер үшін өте қолайлы. Бұл WebRTC архитектурасы қандай болатынын 2.8 суреттен :



2.8 Сурет – Негізгі схемасы

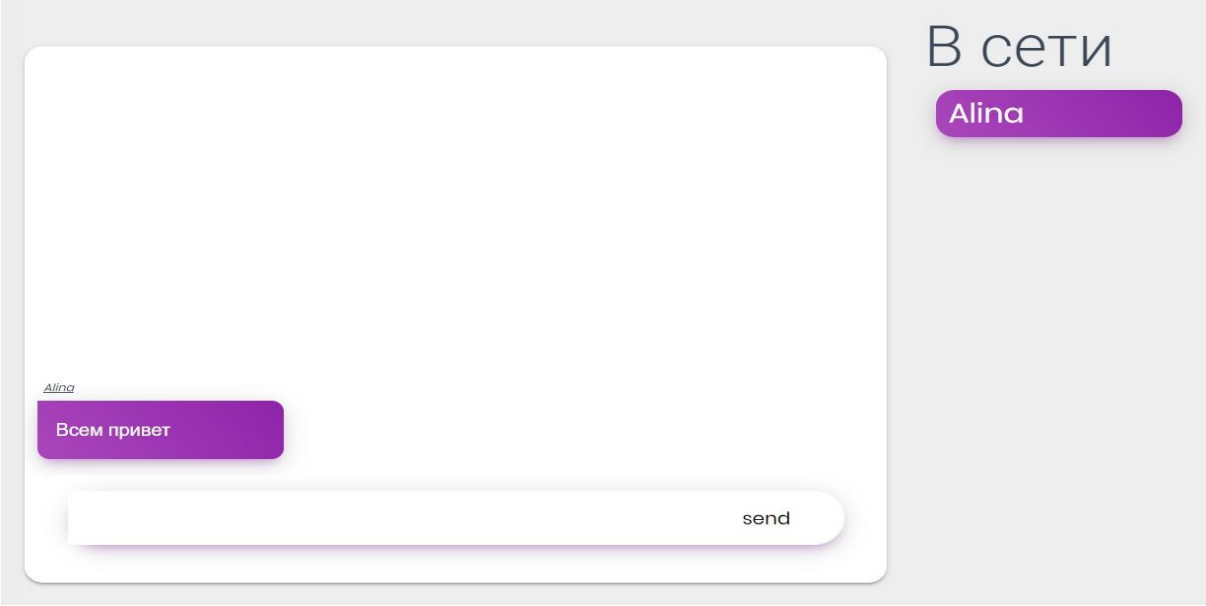
Сигналды сервер WebRTC қосымшасын жүзеге асырудың жалғыз қажет бөлігі емес. Басқа шолғышқа деректерді жіберу және алу үшін клиент белгілі бір ақпаратқа ие болуы керек. Ең маңызды ақпарат - бұл клиенттің жалпыға қол жетімді IP мекенжайы. Басқа клиенттен кіріс қосылымын алу үшін әр клиент немесе шолушы бір-бірі туралы осы ақпаратты білуі керек. Кейде клиенттің жеке IP-мекен-жайын табу қиынға соғады, әсіресе егер клиент брандмауэрдің немесе NAT-нің артында болса, бұл қазіргі кезде өте жиі кездеседі. Мұны жеңу үшін WebRTC STUN серверін пайдалануды ұсынады. STUN сервері клиенттерге өздерінің жеке IP мекенжайын және артта тұрған NAT түрін білуге мүмкіндік береді. NAT-қа байланысты ақпарат алу кейде қиынға соғуы мүмкін, бұл жағдайда TURN сервері бұл процеске көмектесе алады.

STUN арқылы жиналған ақпаратты кейіннен клиенттер арасындағы байланысты үйлестіру үшін Интерактивті қосылымды құру (ICE) -мен бірге қолдануға болады. ICE - бұл WebRTC қолданатын құрылым, және ICE қосылымның қалай жұмыс істейтінін сипаттайды және клиентке «тең-теңімен» байланысын орнату үшін ақпаратты табуға көмектеседі. ICE шешімі - «үміткерлердің» барлық жұптарын жүйелі түрде тестілеу (STUN серверінен алынған ақпарат). ICE адрестердің қайсысы жұмыс істейтінін анықтауға тырысады. Іс жүзінде көптеген ақпарат комбинациясы жұмыс істемейді, сондықтан ICE бұл мәселені шешуге көмектеседі. Енді біз 2.8-суретті екі клиент арасындағы WebRTC байланысы кезінде не болатынын егжей-тегжейлі кеңейте аламыз. Төмендегі 2.9 суретті қараңыз.

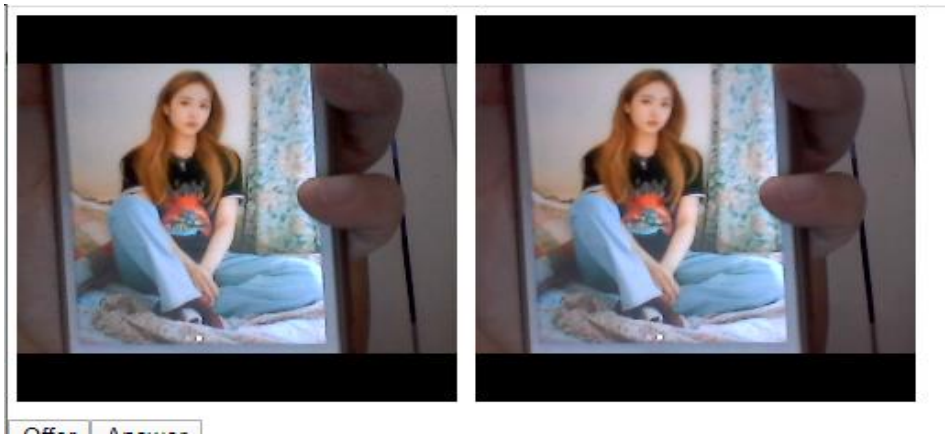


2.9 Сурет – WebRTC интернеттегі байланыс орнатылуы

WebRTC қолдана отырып жасалған чат:



2.10 Сурет – Хабарлама алмасу

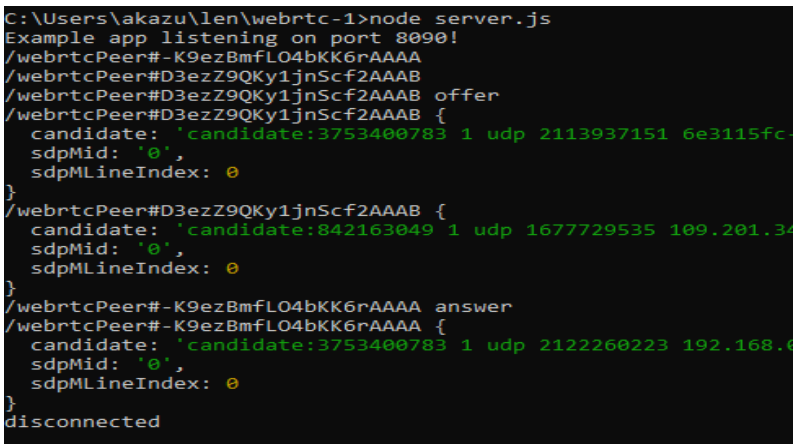


2.11 Сурет – Видео қоңырау

2.11 суретте Видео хабарлама алмасу көрсетілген. Екі қолданушы арасында сокет арқылы хабар алмасу.

2.12 Сурет – Байланыс

2.12 суретте екі қолданушы арасындағы байланыс орнатылуы, яғни олар сокеттер номері арқылы идентфикацияланады да бір-біріне көрсетілгендей хабарламалармен алмасады. Ал келесі суретте егер бір қолданушы шығып кеткен жағдайда:



2.13 Сурет – Байланыс үзілуі

2.13 суретте көрсетілгендей бір қолданушы ажартылып қалды да командалық жолда «disconnected» деп яғни байланыс ажыратылды деп жазылып тұр.

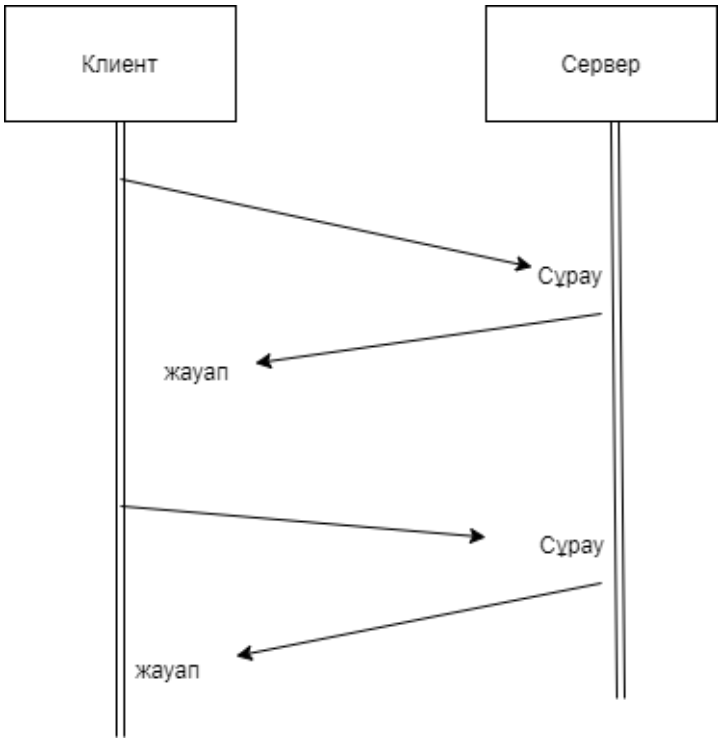
```
  navigator.mediaDevices.getUserMedia(constraints)
    .then(success)
    .catch(failure)
}
```

2.14 Сурет – Кодтың үзіндісі

2.14 сурет - бұл код кішкентай бір бөлігі ғана мысал ретінде көрсетілген браузердің микрофон мен видеокамерасын қолдануға мүмкіндік беретін әдісі.

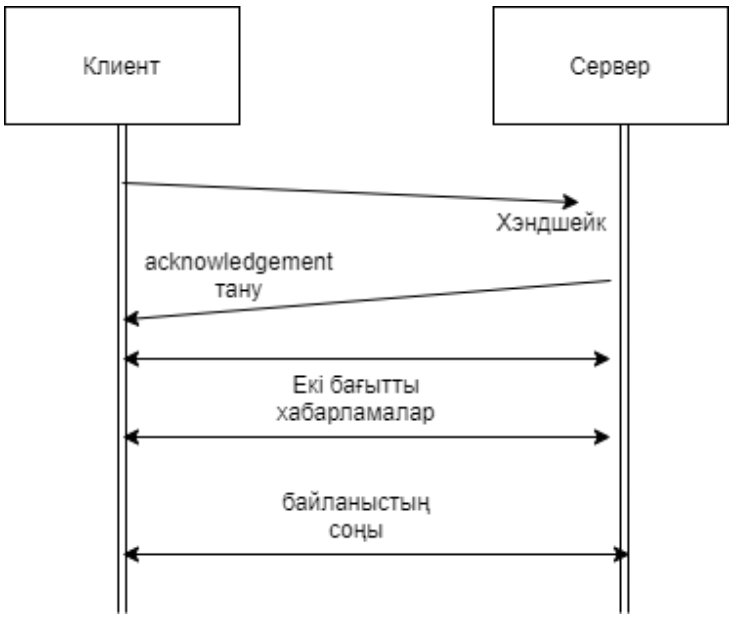
2.2.2 WebSocket протоколы

WebSocket - бұл қос дуплексті байланыс қолдауымен қосымшаның қабаттық протоколы. Ол бір TCP сокетін қолданады, бірақ кейбір негізгі протоколдардың жиектерін қарапайым етеді. WebSocket өнімділік, қарапайымдылық, стандарттар және HTML5 туралы болады. Ол HTTP-мен бірге үздіксіз жұмыс істеуге арналған. WebSocket-тің бірегейлігі неде және оның маңыздылығын түсіну үшін екі бөлікке бөлу керек; протоколдың өзі (RFC 6455) және API.



2.15 Сурет – HTTP қосылыстың өмірлік циклі

WebSocket протоколы басқарылатын ортада сенімсіз кодпен жұмыс істейтін клиенттің сол кодтан байланыс орнатқан қашықтағы хостпен екі жақты байланысын қамтамасыз етеді. Ол үшін пайдаланылатын қауіпсіздік моделі - бұл веб-шолғыштар жиі қолданатын шығу тегі негізіндегі қауіпсіздік моделі. Хаттама TCP арқылы қабатталған негізгі хабарламалар шеңберімен ашылатын қол алмасудан тұрады. Бұл технологияның мақсаты бірнеше HTTP қосылыстарына сенбейтін серверлермен екі жақты байланыс қажет браузерге негізделген қосымшалардың механизмін қамтамасыз ету болып табылады. WebSocket байланысын бастау үшін алдымен HTTP байланысын жасау керек.



2.16 Сурет – WebSocket қосылыстың өмірлік циклі

```
export const websocket = (b) => {
  const socket = new WebSocket("ws://localhost:8080/api/mainchat");

  console.log("Attempting Connection...");

  socket.onopen = () => {
    console.log("Successfully Connected");
    const u = JSON.parse(localStorage.getItem("user_data"));
    socket.send(
      JSON.stringify({ Name: u.username, Text: "sdfs", Register: 1 }
    );
  };

  socket.onmessage = (msg) => {
    console.log(msg);
    b(msg);
  };
```

2.17 Сурет – WebSocket-і қосу

2.17 суретте веб-қосымшасына WebSocket-і құру, сокеттермен байланыс орнатылуы көрсетілген. Бұл хабарлама алмасу, яғни нақты уақыттағы чаттың кодының үзіндісі.

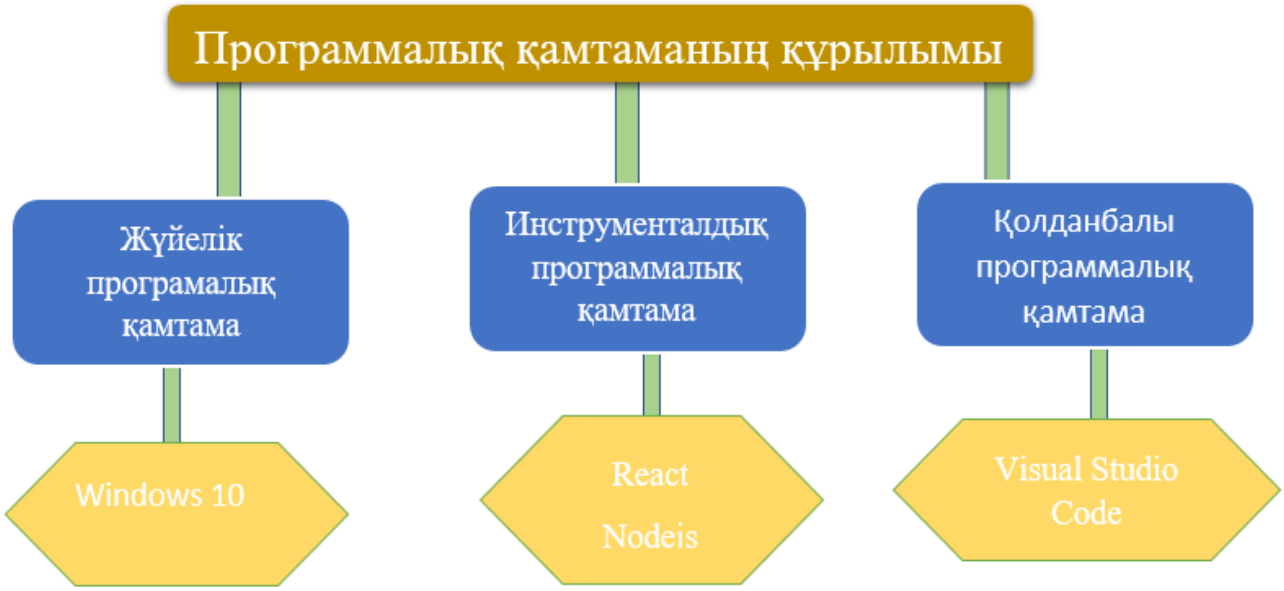
Socket.io - кез-келген платформаға арналған, мысалы, нақты уақыттағы, екі жақты және оқиғалармен алмасуды қамтамасыз ететін браузер. Ол сенімділік пен жылдамдықта бірдей ерекшеленетін бірнеше платформалар арасындағы байланысқа бағытталған. Socket.io - бұл WebSocket-ті іске асыру емес, ол мүмкін болған кезде WebSocket-ті көлік ретінде пайдаланады.

LAMP сияқты веб-бағдарлама стектерін қолдана отырып, нақты уақыттағы деректерді қамтитын қосымшаны жазу, әдетте қиын және қиын. Бұл серверде көптеген конфигурацияларды және сауалнаманы қажет етеді (сауалнама: сыртқы құрылғылардың үлгілерін белсенді түрде бақылайды) және уақыт белгілерін бақылайды. Қосымша байланыс, өңдеу қуаты және көптеген қадамдар қажет. Сондықтан, ол Socket.io-ге қарағанда баяу, ол қажет болған кезде белгілі бір оқиғаны лезде орындайды. Сокеттер нақты уақыттағы байланыс жүйелерінің көпшілігі үшін шешім болып табылады. Клиент пен сервер арасындағы прокси байланыс арқылы екі бағытты байланысты қамтамасыз етуге арналған жүйелер. Сокет - бұл клиент және сервер сияқты екі желілік түйін арасында деректерді жіберуге және алуға арналған ішкі нүкте. Бұл екі платформаның арасында нақты уақыттағы байланыс бар екенін білдіреді, онда сервер оқиғаларды тікелей клиентке жібере алады. Сервер мәліметтерді қабылдайтын және оны барлық қосылған сокеттерге жіберетін динамикалық және интерактивті веб-қосымшаны жасайды. Сондықтан WebSocket протоколға байланысты Socket.io серверіне және керісінше сәтті қосыла алмайды.

Сокет туралы түсінік - оны қолдану үшін пайдалы және пайдалы ететін нәрсе. Socket.IO оқиға туралы дереу жібереді және қабылдайды, прокси-торлар, жүктеме тепе-теңдігі, брандмауэр және антивирустық қосылыстарды орнатқан кезде. Алынған және жаңартылған деректер кейіннен шығару функциясы арқылы сокетке қосылған барлық клиенттерге жіберіледі. Әзірлеуші жобасының құрылымына байланысты шығаратын функциялардың бірнеше түрлері бар. Қосылым Engine.io-ға сүйенеді, ол ұзақ сауалнамамен байланыс орнатуға тырысады. Байланыс орнатылғаннан кейін деректерді жылдамырақ беру үшін WebSocket қосылымын жаңартуға тырысады.

3 ВЕБ-ҚОСЫМШАНЫҢ ҚҰРЫЛЫМЫ

3.1 Проргаммалық қамтаманың құрылымы



3.2.1 Жалпы мағлұмат

Бұл бағдарлама React кітапханасын, WebRTC технологиясын, сокеттерді қолдану арқылы жасалған. Орындалуы үшін Microsoft Windows 10 графикалық операциялық жүйесі және Visual Studio Code, node.js, программалары қажет.

3.2.2 Функционалдық тағайындалуы

Құрылған қосымша келесілерді қамтамасыз етеді:

Қолданушылар арасындағы нақты уақыттағы байланыс, интерактивтілік, ақпарат алмасу.

Есептелулер: Мәліметтер базасындағы берілген мәліметтерді шығарады.

3.2.3 Логикалық құрылымның баяндалуы

Каталогтағы ойындарды шығару;

Қолданушылардың ойынға қосылуы;

Қолданушылар ақпарат алмасуы хабарлама арқылы немесе видео-байланыс арқылы.

3.2.4 Шақыру және жүктеу

Қосымшаны шақыру үшін, бірінші консольда React кітапханасында жазылған клиенттік API-ді “npm start” командасы арқылы шақырамыз, ол үшін Node.js орнатылуы қажет.

Екіншіден, Go тілінде жазылған серверді шақырамыз.

3.2.5 Қолданылған техникалық құралдар

Бұл Дипломдық жобаны жазғандуға келесі құралдар қолданылды:

– Ноутбук HP X540LJ, Intel(R) Core(TM) i5-9700HF CPU @ 2.4GHz, 8.00 ГБ ЖАД.

– Алынбалы флэш-жинақтағыш 32 ГБ USB 3.1.

– Принтер Samsung dj 480c.

3.2.6 Кіріс және шығыс мәлімет

Кіріс мәліметтер ретінде клиенттердің ойын барысындағы өзгерістер, хабарламалар.

Шығыс мәліметтерге қолданушы енгізілген мәліметтер бойынша рейтинг жасау, сақталынған ойындар, оларға керекті мәліметтерді іздеу жүйесі бойынша шығару.

ҚОРЫТЫНДЫ

Соңғы жылдары веб-қосымшалардың дамыған кезеңі деп айтсақ болады. Алайда Қазақстанда ол әлі де дамуының, өсуінің орыны бар. Себебі қазіргі орын алған жағдайға қарап интернет желісінің осал жерлері бар екеннің байқадық. Сол себепті барлық салаларды барынша онлайн режиміне ауысуы заманына аттанғалы тұрмыз. Осы дипломдық жобаның да негізгі мақсаты нақты уақыттағы веб-қосымша құру болатын. Соған байланысты қолданушыға ыңғайлы интерфейс таңдалып, нақты уақытта хабар алмасу технологиялары туралы мәмлеге келу.

Дипломдық жобының нәтижесі ретінде нақты уақытта жұмыс жасайтын веб-қосымша құрылды. Осы жобаның негізінде білім беру саласында, бизнес конференциялар басқада салаларға енгізуге болады. Нақты уақыттағы технологиялардың қазіргі және болашақта алатын орны зор болмақ.

Қысқартулар тізімі

API - Application Program Interface
DOM - Document Object Model
JS - JavaScript
JSX - JavaScript XML
URL - Uniform Resource Locator
HTML - Hyper Text Markup Language
ICE - Interactive Connectivity Establishment
NAT - Network Address Translation
RTC - Real Time Communication
SDP - Session Description Protocol
STUN - Simple Transversal Utilities for NAT
TURN - Traversal Using Relays around NAT
UDP - User Datagram Protocol
WebRTC - Real Time Communications for the Web
WWW - World Wide Web

ПАЙДАЛАНЫЛҒАН ӘДЕБИЕТЕР ТІЗІМІ

1 Стефанов Стоян «React: Up and Running: Building web Applications. First Edition», 2016.

2 Хортон Адам, Вице Раяйн « Мастерим Реакт», 2016.

3 А.Хейккинен, Т.Коскела, М. Илианттила, «Performance evaluation of distributed data delivery on mobile devices using WebRTC», 2015.

4 М. Адейя, И. Макитла, Т. Фогуил, «Determining the signalling overhead of two common WebRTC methods: JSON via XMLHttpRequest and SIP over WebSocket» in AFRICON, 2013.

5 М.Пацианский «React.js для начинающих», 2016.

6 Алекс Янг, Бредли Мек, Майк Кантелон «Node.js в действии», 2017.

7 Марк Тиленс Томас «React в действии», 2017.

8 «WebRTC Home,» WebRTC. [Online]. Available: https://webrtc.org/. [Accessed: 25-Jan2019].

9 BasicsofWebRTCpeerconnection. //Электронды ресурс: https://medium.com/programming-ideas-tutorial-and-experience/basics-of-webrtc-peer-connections-c1ed743de2f6.

А Қосымшасы

```
const express = require('express')
var io = require('socket.io')
({
  path: '/webrtc'
})
const app = express()
const port = 8090
app.use(express.static(__dirname + '/build'))
app.get('/', (req, res, next) => {
   res.sendFile(__dirname + '/build/index.html')
})
const server = app.listen(port, () => console.log(`Example app listening on port
${port}!`))
io.listen(server)
const peers = io.of('/webrtcPeer')
let connectedPeers = new Map()
peers.on('connection', socket => {
  console.log(socket.id)
  socket.emit('connection-success', { success: socket.id })
  connectedPeers.set(socket.id, socket)
  socket.on('disconnect', () => {
    console.log('disconnected')
    connectedPeers.delete(socket.id)
  })
  socket.on('offerOrAnswer', (data) => {
    for (const [socketID, socket] of connectedPeers.entries()) {
       if (socketID !== data.socketID) {
        console.log(socketID, data.payload.type)
        socket.emit('offerOrAnswer', data.payload)
       }
    }
  })
  socket.on('candidate', (data) => {
       for (const [socketID, socket] of connectedPeers.entries()) {
      if (socketID !== data.socketID) {
        console.log(socketID, data.payload)
        socket.emit('candidate', data.payload)
      }
    }
  })
})
```

Б Қосымшасы

```
import React, { Component } from 'react';
import io from 'socket.io-client'
class App extends Component {
  constructor(props) {
    super(props)
    this.localVideoref = React.createRef()
    this.remoteVideoref = React.createRef()
    this.socket = null
    this.candidates = []
  }
  componentDidMount = () => {
    this.socket = io(
      '/webrtcPeer',
      {
        path: '/webrtc',
        query: {}
      }
    )
    this.socket.on('connection-success', success => {
      console.log(success)
    })
    this.socket.on('offerOrAnswer', (sdp) => {
      this.textref.value = JSON.stringify(sdp)
          this.pc.setRemoteDescription(new RTCSessionDescription(sdp))
    })
    this.socket.on('candidate', (candidate) => {
      this.pc.addIceCandidate(new RTCIceCandidate(candidate))
    })
    const pc_config = {
      "iceServers": [
        {
          urls : 'stun:stun.l.google.com:19302'
        }
      ]
    }
this.pc = new RTCPeerConnection(pc_config)
    this.pc.onicecandidate = (e) => {
     if (e.candidate) {
        this.sendToPeer('candidate', e.candidate)
      }
    }
    this.pc.oniceconnectionstatechange = (e) => {
```

```
  console.log(e)
  }
    this.pc.ontrack = (e) => {
    this.remoteVideoref.current.srcObject = e.streams[0]
  }
  const success = (stream) => {
    window.localStream = stream
    this.localVideoref.current.srcObject = stream
    this.pc.addStream(stream)
  }
  const failure = (e) => {
    console.log('getUserMedia Error: ', e)
  }
  const constraints = {
    audio: false,
    video: true,
        }
  navigator.mediaDevices.getUserMedia(constraints)
    .then(success)
    .catch(failure)
}
sendToPeer = (messageType, payload) => {
  this.socket.emit(messageType, {
    socketID: this.socket.id,
    payload
  })
}
createOffer = () => {
  console.log('Offer')
  this.pc.createOffer({ offerToReceiveVideo: 1 })
    .then(sdp => {
      this.pc.setLocalDescription(sdp)
      this.sendToPeer('offerOrAnswer', sdp)
  })
}
createAnswer = () => {
  console.log('Answer')
  this.pc.createAnswer({ offerToReceiveVideo: 1 })
    .then(sdp => {
      this.pc.setLocalDescription(sdp)
      this.sendToPeer('offerOrAnswer', sdp)
  })
```

Б қосымшасының жалғасы

```
        }
        setRemoteDescription = () => {
          const desc = JSON.parse(this.textref.value)
          this.pc.setRemoteDescription(new RTCSessionDescription(desc))
        }
        addCandidate = () => {
          this.candidates.forEach(candidate => {
            console.log(JSON.stringify(candidate))
            this.pc.addIceCandidate(new RTCIceCandidate(candidate))
          });
        }
        render() {
          return (
            <div>
              <video
                style={{
                  width: 240,
                  height: 240,
                  margin: 5,
                  backgroundColor: 'black'
                }}
                ref={ this.localVideoref }
                autoPlay>
              </video>
              <video
                style={{
                  width: 240,
                  height: 240,
                  margin: 5,
                  backgroundColor: 'black'
                }}
                ref={ this.remoteVideoref }
                autoPlay>
              </video>
              <br />
              <button onClick={this.createOffer}>Offer</button>
              <button onClick={this.createAnswer}>Answer</button>
              <br />
              <textarea style={{ width: 450, height:40 }} ref={ref => { this.textref = ref
}} />
            </div>
          )
```

```
 }
}
export default App;
```